IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

REPLY BRIEF

Applicant:	Barsness, *et al.*	Docket No.:	ROC920030264US1
Serial No.:	10/666,032	Group Art Unit:	2193
Filed:	09/18/03	Examiner:	Mitchell, Jason D.

TITLE: INTER-JOB BREAKPOINT APPARATUS AND METHOD

Mail Stop APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

This Reply Brief is filed to address the arguments in the Examiner's Answer dated 03/31/2008.

ARGUMENT

Issue 1: Whether claims 8, 17, and 28 are unpatentable under 35 U.S.C. §103(a) as being obvious in view of Haban and Heinen.

Claims 8, 17, and 28

Appellant stands on the arguments made in the Appeal Brief with respect to claims 8, 17 and 28, which are incorporated herein by reference. In addition, appellant adds the remarks below. In the Examiner's Answer, the examiner states at p. 7 first full paragraph:

> Secondly, Haban teaches examples of breakpoints wherein variables are consulted and manipulated in order to send a message (see p. 166, Introduction and pg. 2.1, Primitive Events). Those of ordinary skill in the art would have recognized that manipulating and monitoring of variables by a process that execute [sic] separately and do not share memory and thus relays related messages between one another, to illustrate modify of a program variable by at least (1) the definition of a variable or the (2) functions or instructions of the process to the variable (i.e. assigning variables function Table 1).

The examiner's reading of Haban is incorrect. Table 1 of Haban teaches "assign variable <varID>" as a possible primitive event. A global event in Haban is composed of one or more primitive events or previously defined global events. Haban at p. 166, second column, last paragraph at lines 13-15. The "assign variable" primitive is a "Program Level" primitive shown in Table 1. This means a global event could be defined based on the "assign variable" primitive referenced in Table 1, which would mean an action would be performed when the "assign variable" global event is satisfied. Note the "assign variable" global event is satisfied when an instruction in a program being monitored assigns a value to a variable within the program. A correct reading of Haban is that Haban may monitor a program for when the program assigns a value to one of its

own variables. The assigning of a value by a program to one of its own variables has nothing to do with modifying a program variable in a second job, as recited in claim 8. In addition, the assigning of a value by a program to one of its own variables has nothing to do with "manipulating and monitoring of variables by a process that execute [sic] separately", as stated by the examiner. For this reason, the examiner's characterization of Haban is incorrect. While Haban teaches a program level primitive when a program assigns a program variable, Haban has no teaching of any process modifying the value of a variable in any other process.

The examiner then goes on in the second full paragraph of p. 7 of the Examiner's Answer to attempt to justify that the claim language could include manual operation or automated operation. This discussion misses the point and clouds the issues. Claim 8 recites "an inter-job breakpoint mechanism that detects at least one condition in the first job, and in response thereto, modifies a program variable in the second job." The examiner seems to be overlooking the "in response thereto" phrase in the claim. Whether the mechanism in claim 8 is hardware, software, or some combination, it detects at least one condition in the first job, and ***in response thereto***, modifies a program variable in the second job. Using the examiner's own logic, if Haban detects a global event, and then allows a user to modify a program variable in a second job as taught in Heinen, this still does not read on the limitations in the pending claims because the modification of a program variable by a user using the DEPOSIT command in Heinen is not done ***in response to*** the mechanism in Haban detecting the global event. Even combining Haban and Heinen as suggested by the examiner fails to read on all of the limitations in the claims because the modification of a program variable by a user using the DEPOSIT command as taught in Heinen is not performed ***in response to*** the mechanism in Haban detecting the global event. As a result, the examiner's rejection of claims 8, 17 and 28 under 35 U.S.C. §103(a) based on the combination of Haban and Heinen is in error.

The examiner attempts to shore up his rejection by citing to appellant's specification at p. 15 lines 8-10. This attempt in ineffective. The general language in appellant's specification does not water down the express language in the claims of an inter-job breakpoint mechanism that modifies a program variable in the second job ***in response to*** the mechanism detecting at least one condition in the first job. Note it is the same inter-job breakpoint mechanism that 1) detects at least one condition in the first job, and 2) in response thereto, modifies a program variable in the second job. The examiner states: "Heinen's disclosure of a user command does not 'teach away' from the claim since the claims are broad in scope and can be read in multiple manners." Appellant reiterates the user command in Heinen expressly teaches away from the limitations in the pending claims because a user using a DEPOSIT command to change a variable in Heinen does not read on a single mechanism in the claims that that 1) detects at least one condition in the first job, and 2) ***in response thereto***, modifies a program variable in the second job. Appellant respectfully asserts the examiner is reading the limitations in the pending claims beyond their broadest reasonable interpretation because the pending claims clearly recite a single mechanism that both detects at least one condition in the first job, and ***in response thereto***, modifies a program variable in the second job. Nowhere does Haban, Heinen nor their combination teach or suggest the single mechanism in the claims that performs both of these functions. The examiner's hand waving is ineffective in addressing the defects in the examiner's rejection because the mechanism in Haban that detects a global event is a different mechanism than the mechanism in Heinen that allows a user to manually modify a program variable in the second job using a DEPOSIT command. A reasonable combination of Haban and Heinen would allow a system to halt upon the occurrence of a global breakpoint as taught in Haban, followed by a user manually executing a DEPOSIT command to modify a program variable in a different process. Such a combination of Haban and Heinen, even if proper, does not read on a single mechanism as recited in the claims that 1) detects at least one condition in the first job, and 2) ***in response thereto***, modifies a program variable in the second job.

In response to appellant's contention that the combination of Haban and Heinen to arrive at the limitations in the claims is only possible through the use of impermissible hindsight reconstruction, the examiner for the first time in the Examiner's Answer provides a detailed rationale for the combination. Appellant stands on the arguments made in the Appeal Brief regarding the lack of motivation to combine Haban and Heinen. The stated rationale for the combination in the final office action was "to provide means for debugging distributed processes", which does not address why one of ordinary skill in the art would be motivated to combine Haban and Heinen in the manner claimed. The examiner should be precluded from providing a detailed rationale for the combination for the first time in the Examiner's Answer when the rationale for the combination in the final rejection is fatally flawed because it does not address why one of ordinary skill in the art would be motivated to combined Haban and Heinen in the manner claimed. As a result, applicant reiterates the arguments in the Appeal Brief that the examiner failed to state a valid rationale for combining Haban and Heinen in the manner claimed, and therefore failed to establish a prima facie case of obviousness under 35 U.S.C. §103(a).

For the many reasons given above, the examiner's rejection of claims 8, 17 and 28 is in error. Appellant respectfully requests the examiner's rejection of claims 8, 17 and 28 under 35 U.S.C. §103(a) be reversed.

Issue 2: **Whether claims 9, 18, and 29 are unpatentable under 35 U.S.C. §103(a) as being obvious in view of Haban and Heinen in view of Diec.**

Claims 9, 18, and 29

Appellant stands on the arguments made in the Appeal Brief with respect to claims 9, 18 and 29, which are incorporated herein by reference. In addition, appellant adds the remarks below.

The examiner attempts in the Examiner's Answer to clear up the ambiguity in the examiner's rejection of claims 9, 18 and 29 as identified in the Appeal Brief. This attempt is ineffective. The trace data in Diec is stored in a logfile. The logfile in Diec has nothing to do with a "second job's output" as recited in claim 9, and therefore the writing of trace data in Diec to a logfile does not read on "outputs a debug message to the second job's output." The examiner states at p. 11 of the Answer:

> Accordingly, Diec's teachings, in combination with the disclosure of Haban read on the broad interpretation of the limitations, specifically Haban's inter-job breakpoint detects met conditions and output (by sending Diec's message) a debug message (Diec's trace data) to a second job (Haban's second job) for later storage in an output file system.

This language by the examiner shows the fatal flaw in the rejection. Claim 9 recites "outputs a debug message to the second job's output." A reasonable combination of Haban and Diec might include the generation of a global breakpoint in Haban, followed by the generation of a message as taught in Diec to generate trace data as taught in Diec. Note, however, the trace data in Diec, although it is later stored, is not output "to the second job's output" as recited in claim 9. As a result, the examiner's rejection of claim 9 based on the combination of Haban, Heinen and Diec is in error.

The examiner apparently recognized the weakness in the rejection of claim 9 in the final office action because the examiner provided for the first time in the Examiner's Answer a new rationale for the rejection. In doing so, the examiner recites a definition for Wikipedia of "output buffer", then posits it would have been obvious to one or ordinary skill in the art to "deposit data" into an output buffer, thus meeting the claim limitation "outputs a message to the second job's output." The examiner thus piles one bad assumption on top of another. None of the three cited references teach an output buffer. Yet the examiner pulls the output buffer out of thin air via a reference to Wikipedia, and states it would have been obvious to those of ordinary skill in the art to use the DEPOSIT command in Heinen to output to some hypothetical output buffer for the second job in the claim. Appellant notes the examiner simply stated "could be used by those of ordinary skill in the art", and did not state it would have been obvious to one of ordinary skill of the art at the time the invention was made. Furthermore, the examiner failed to provide any rationale for this combination other than the examiner's creative imagination. As a result, this new grounds of rejection raised for the first time in the Examiner's Answer is defective, and the examiner has failed to establish a prima facie case of obviousness for claim 9 under 35 U.S.C. §103(a).

Appellant further stands on the detailed arguments made on p. 8-9 in the Appeal Brief regarding the inconsistencies in the rejection of claim 9 based on the combination of Haban, Heinen and Diec.

On p. 12 of the Examiner's Answer, the examiner attempts to provide for the first time a detailed rationale for combining Haban, Heinen and Diec. This attempt to introduce a detailed rationale for the combination for the first time in the Examiner's Answer should be rejected by the Board. The final rejection of claim 9 does not include motivation for combining Haban, Heinen and Diec in the manner claimed. Because the examiner's motivation to combine is a general benefit, and does not address reasons why one of ordinary skill in the art would combine Haban, Heinen and Diec in the manner

claimed, the examiner has failed to establish a prima facie case of obviousness for rejecting claim 9 under 35 U.S.C. §103(a).

A reasonable combination of Haban, Heinen and Diec in a manner consistent with their teachings would result in the ability to start logging trace data for a software application as taught in Diec in the system of Haban-Heinen. Nowhere is there any suggestion in any of Haban, Heinen, Diec nor their combination that would suggest to one of ordinary skill in the art to combine these references in the manner recited in claim 9. For these reasons, claim 9 is allowable over the combination of Haban, Heinen and Diec.

Claims 18 and 29 include limitations similar to those in claim 9 addressed above, and are therefore allowable for the same reasons. In addition, claims 9, 18 and 29 are also allowable as depending on allowable independent claims. Appellant respectfully requests the examiner's rejection of claims 9, 18 and 29 under 35 U.S.C. §103(a) be reversed.

CONCLUSION

Claims 8-9, 17-18, and 28-29 are addressed in this Appeal. For the numerous reasons articulated above, appellant maintains that the rejections of claims 8-9, 17-18, and 28-29 are erroneous.

Appellant respectfully submits this Reply Brief fully responds to, and successfully contravenes, the arguments in the Examiner's Answer and respectfully requests that the final rejection be reversed and that all claims in the subject patent application be found allowable.

Respectfully submitted,

By /derekpmartin/
Derek P. Martin
Reg. No. 36,595

MARTIN & ASSOCIATES, L.L.C.
P.O. Box 548
Carthage, MO 64836-0548
(417) 358-4700
Fax (417) 358-5757

CLAIMS APPENDIX

1-7 (Cancelled)

8. An apparatus comprising:

at least one processor;

a memory coupled to the at least one processor;

a first job residing in the memory and executed by the at least one processor;

a second job residing in the memory and executed by the at least one processor;

an inter-job breakpoint mechanism that detects at least one condition in the first job and, in response thereto, modifies a program variable in the second job.

9. The apparatus of claim 8 wherein the inter-job breakpoint mechanism, in response to detecting the at least one condition in the first job, outputs a debug message to the second job's output.

10-16 (Cancelled)

17. A method for debugging comprising the steps of:

defining at least one condition in a first job;

defining at least one action to take on a second job;

monitoring execution of the first job;

monitoring execution of the second job; and

when the at least one condition in the first job is satisfied, modifying a program variable on the second job.

18. The method of claim 17 further comprising the step of outputting of a debug message to the second job's output when the at least one condition in the first job is satisfied.

19-27 (Cancelled)

28. A computer-readable program product comprising:

(A) an inter-job breakpoint mechanism that monitors execution of first and second jobs, and when at least one condition in the first job is satisfied, modifies a program variable on the second job; and

(B) recordable media bearing the inter-job breakpoint mechanism.

29. The program product of claim 28 wherein the inter-job breakpoint mechanism, in response to detecting the at least one condition in the first job, outputs a debug message to the second job's output.

30-32 (Cancelled)